Video2Vec: Learning Semantic Spatio-Temporal Embedding for Video Representations

by

Sheng-Hung Hu

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2016 by the
Graduate Supervisory Committee:

Baoxin Li, Chair
Pavan Turaga
Jianming Liang
Hanghang Tong

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

High-level inference tasks in video applications such as recognition, video retrieval, and zero-shot classification have become an active research area in recent years. One fundamental requirement for such applications is to extract high-quality features that maintain high-level information in the videos.

Many video feature extraction algorithms have been purposed, such as STIP, HOG3D, and Dense Trajectories. These algorithms are often referred to as "handcrafted" features as they were deliberately designed based on some reasonable considerations. However, these algorithms may fail when dealing with high-level tasks or complex scene videos. Due to the success of using deep convolution neural networks (CNNs) to extract global representations for static images, researchers have been using similar techniques to tackle video contents. Typical techniques first extract spatial features by processing raw images using deep convolution architectures designed for static image classifications. Then simple average, concatenation or classifier-based fusion/pooling methods are applied to the extracted features. I argue that features extracted in such ways do not acquire enough representative information since videos, unlike images, should be characterized as a temporal sequence of semantically coherent visual contents and thus need to be represented in a manner considering both semantic and spatio-temporal information.

In this thesis, I propose a novel architecture to learn semantic spatio-temporal embedding for videos to support high-level video analysis. The proposed method encodes video spatial and temporal information separately by employing a deep architecture consisting of two channels of convolutional neural networks (capturing appearance and

local motion) followed by their corresponding Fully Connected Gated Recurrent Unit (FC-GRU) encoders for capturing longer-term temporal structure of the CNN features. The resultant spatio-temporal representation (a vector) is used to learn a mapping via a Fully Connected Multilayer Perceptron (FC-MLP) to the *word2vec* semantic embedding space, leading to a semantic interpretation of the video vector that supports high-level analysis. I evaluate the usefulness and effectiveness of this new video representation by conducting experiments on action recognition, zero-shot video classification, and semantic video retrieval (word-to-video) retrieval, using the UCF101 action recognition dataset.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Dr. Baoxin Li for his wise advice and guidance through the process of researching and writing this thesis. His expertise and deep insights in the field of Computer Vision and Deep Learning help me to discover my weaknesses and steered me to the right direction.

I would also like to express my profound gratitude to my committee members Dr. Pavan Turaga, Dr. Hanghang Tong and Dr. Jianming Liang for their time and attendance to validate this research project.

I would also like to acknowledge my research partner and lab mate, Mr. Yikang Li for helping me and inspiring me with novel ideas, resolving experimental issues and designing algorithms. We two made an awesome team!

I appreciate the support from all my lab-mates for keeping the work environment friendly and comfortable and for their valuable comments on this work.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Author

Sheng-Hung Hu

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION:

Many computer vision applications involve general scene understanding based on videos. Examples include video-based action/event recognition, vision for human-computer interaction, and video surveillance, etc. One fundamental task in these applications is to establish a certain mapping from a raw video input to some high-level semantic labels such as action or event categories, or gesture-based commands, etc. Typically, raw video data would first be processed for feature extraction before any technique for establishing the above mapping is applied. The quality of the features, or more generally, the representation of the videos, plays an important role and can have significant impacts on subsequent analysis tasks.

Some well-known video features include STIP [1], HOG3D [2], and Dense Trajectories [3], all having been widely used in video-based analyses. These are often called "handcrafted" features, since they were deliberately designed based on reasonable considerations. However, handcrafted features are often expensive to extract and perform poorly when dealing with complex scenes or conducting high-level vision tasks. In contrast, techniques relying on deep neural networks for directly learning features from videos have been seen in recent years. For example, following the success of convolutional neural network (CNN) for image classification [4], a CNN-based architecture was proposed to fuse appearance and optical flow features to form a (frame-level) video descriptor in [5]. Typically, features learned from such approaches are based

on the output of the fully connected layer in a CNN that contains many hidden layers acting as progressive feature extractors. Compared with handcrafted features, CNN-based architectures have advantages on feature extracting speed and can be trained to suit specific purposes by applying different loss functions.

While frame-level CNN-based methods outperform most handcrafted features in both complexities and accuracies, a CNN-based approach like [5] does not have the capacity to model the global temporal evolution of the video/features, which may be the key to higher-level semantic analysis. A naive approach of averaging frame-level representations to form a global representation would not solve the issue, as the temporal structure is no longer maintained. Furthermore, CNN-based methods, although rich in spatial visual information, still lack a semantic organization or clustering that would directly facilitate a higher-level analysis task like action labeling. Therefore, a semantic spatio-temporal embedding for video representation is highly desired.

1.2 LITERATURE REVIEW:

To create such embedding, I need to globally encode information in all semantic, spatial and temporal manners for a given video. Thus, I will briefly review some state-of-the-art embedding and feature extraction methods for linguistic, spatial and temporal respectively in the following paragraph.

1.2.1 Linguistic Embedding:

Linguistic embedding algorithms aim to capture human semantic knowledge and map to vectors of real numbers in a low-dimensional space relative to the vocabulary size.

Such works include word-level embedding, sentence-level embedding and document-level embedding. In this work, I focus on word-level embedding.

Bengio et al [6] first introduce a method to reduce the high-dimensionality of words in context by modeling a distributed representation for the target word in 2003. In [6], they first represent a statistical model of language by the conditional probability of the next word given the previous context (a bag of words) as:

$$\hat{p}(w_1^T) = \prod_{t=1}^{T} \hat{p}(w_t \mid w_1^{t-1}), \tag{1}$$

where w_t is the t-th word, and $w_i^j = (w_i, w_{i+1}, \cdots, w_{j-1}, w_j)$ is the writing sub-sequence. Then, they initial a word by a word feature vector (real-valued vector), use the conditional probability model to represent word sequence and finally learn simultaneously the word feature vector and the model parameters by maximizing Equation 1. Their methods are often referred to as the *n-gram* models as they often consider only the previous *n* (*n* as a pre-defined parameter) words to avoid over general representation. That is:

$$\hat{p}(w_t \mid w_1^{t-1}) \approx \hat{p}(w_t \mid w_{t-n+1}^{t-1}). \tag{2}$$

In 2010, Mikolov et al purposed an efficient word embedding method called *word2vec* [7]. In this work, they introduced four distinct shallow training models – Continuous Bag-of-Words (CBOW) with or without negative sampling and skip-gram with or without negative sampling. With outstanding efficiency, *word2vec* can be trained on large-scale dataset — hence more general and can include a relative large dictionary size. I will discuss the *word2vec* in more detail in later chapter as it serves as the basic building blocks in this research project.

1.2.2 Spatial Embedding:

Spatial embedding methods aim to explore both local and global information in the two-dimension spatial domain and map to a real value vector. Such works include image descriptors, geometric analysis and graphs analysis. In this work, I focus on image descriptors.

Conventional image descriptors, such as SIFT [8] and SURF [9] present images by describing each pixel (thus called dense local features) by its relationships with surrounding pixels according to a pre-designed manner. The extracted features are robust, scale and rotation invariant (SURF) when dataset is relatively small but weak against large-scale dataset with large variants (such as ILSVRC ImageNet challenge [10]). Also, due to its pre-designed manner, conventional image descriptors often suffered from lack of high-level representing abilities.

In recent years, deep Convolution Neural Networks (CNNs) have been widely applied in tackling this challenge thanks to the fast growth in GPU/parallel system development. Architectures such as AlexNet [4], Google-inception models [11] and VGG-net [12] have achieved great success in ImageNet classification challenge. Deep CNN approaches expand the "receptive field" and simultaneously enhance representing abilities against highly non-linear objects by stacking multiple convolution operations and down-sample (pooling) images – thus obtain high-level embedding of both local and global spatial information. Furthermore, Deep CNN approaches can be trained to tackle against a specific task by deploying different loss functions. In this work, I select VGG-16 layer network as basic building blocks to extract spatial information per video frame.

1.2.3 Temporal Embedding:

Temporal embedding methods aim to exhibit dynamic behavior in the temporal domain. Such works include financial model prediction, speech processing, handwriting recognition, object tracking and motion tracking. In this work, I focus on motion tracking.

Perhaps the most well-known convention temporal embedding methods for motion tracking are the optical flow methods. The optical flow methods try to calculate the motion between two (or more) image frames, which are taken at times t and $t + \Delta t$ at every pixel position. Dense optical flow can very well capture the motion between two frames pixel-wisely but fail to encode the global temporal information (produce one optical flow per frame). Also, optical flow methods are rarely expensive in computation.

Learning-based methods have become more and more popular in this domain as well. Recurrent Neural Networks (RNNs) are applied to solve this challenge. Generally, RNN architectures contain three different nodes – input nodes, output nodes and hidden nodes. When encoding, each frame in the video serves as an input signal through input nodes. Then, the hidden nodes are applied to the inputs to serve as temporal filters. Finally, output nodes produce the encoded signal. The RNNs-embedded signals are capable of capturing global temporal information with less complexity and hence more desired in video analysis field. However, same as other deep learning structures, RNNs often suffered from the gradient vanishing problem once the input sequence goes longer. Therefore, gated alternatives of RNNs are developed; they avoid the gradient vanishing problem by implemented "gating units" that decide what to "forget" and what to "memorize". Two widely used of the gated alternatives of RNNs are the Long-Short

Term Memory (LSTM) [13] and Gated Recurrent Unit (GRU) [14]. I will introduce these two alternatives and RNN in detail in later section as they serve as the basic building blocks in this work.

1.2.4 Spatio-Temporal Embedding:

Conventional action recognition tasks utilize handcrafted descriptors such as STIP, HOG3D, and Dense Trajectories [1]–[3] to capture spatio-temporal information. Such features have found wide applications in the literature. However, in general, handcrafted dense features lack semantic and discriminative capacity and thus cannot effectively represent higher-level information [15]. Recent years have seen deep-learning approaches to video feature extraction. Simonyan et al [5] introduced a method to fuse both CNN appearance and optical flow features while Wang et al [15] proposed descriptors that fuse both low-level handcrafted features and CNN features by using Support Vector Machine to represent videos.

Despite of the fusion process, the above deep-learning approaches still do not fully leverage temporal information of the given video. To overcome this limitation, several more recent approaches [16]–[18] attempted to encode the video by LSTM. LSTM can be considered as a gated RNN, which is capable of discovering the implicit temporal structure of the input sequences while avoiding the gradient vanishing problem. In the above literature, representing videos by LSTM was shown to have some advantages when modeling complex temporal dynamics and competitive results on tasks like action recognition have been reported. Unfortunately, all of the above LSTM-based methods encode videos without considering semantic meanings of the representation. As

a result, the learned representation does not directly support high-level semantic tasks such as semantic video retrieval (retrieving videos by word descriptions never used in training) and zero-shot video classification (recognizing unseen video categories).

Associating video representations with semantics has been studied in various contexts including content-based video retrieval [19]–[22]. In [17], [23], [24], attempts have been made to generate semantic label sequences from video inputs. However, these efforts do not seem to explicitly associate videos with semantic meaning derived directly from the semantic labels of the videos (but rather relying on external dictionaries). Lacking is a learned embedding of videos that may directly lead to semantic interpretation of a novel video, which may or may not have any textual labels. Such an embedding could lead to a new presentation that supports high-level semantic analysis. My work in this project attempts to achieve this by learning the mapping between spatio-temporal representations and the label vectors in the *word2vec* semantic embedding space (SES).

1.3 CONTRIBUTIONS:

In the following paragraph, I will briefly introduce three high-level vision tasks, which the proposed semantic spatio-temporal embedding video representations can fully apply on. They are the video action recognition, the zero-shot classification and the semantic video retrieval. The result, benchmark comparison and discussion for the proposed model conducting these three tasks are shown in later chapter.

1.3.1 Video Recognition:

In this project, I employed the proposed semantic embedding for the task of video recognition on the UCF101 Action Recognition dataset [25] to serve as a proof point. The representation yielded competitive accuracy with the current state-of-the-art method.

1.3.2 Zero-shot Classification:

The proposed semantic embedding will easily support zero-shot learning to classify unseen categories. Zero-shot classification has always been treated as one of the most challenge task in video analysis since the lacking of trained references (labeled examples) for test videos. Most existing zero-shot classification techniques focus on static images and many rely on attributed-based representations [26]. In practice, it is difficult to obtain sufficient amount of data for training comprehensive attribute-based representations for a large number of categories. My representation is advantageous on this regard since the basic embedding space derives its semantics from general human knowledge base (e.g., meanings of words learned from *Wikipedia* documents), and only the last stage of mapping requires video labels to train.

1.3.3 Semantic Video Retrieval:

As an extension to the above work, the embedding representations are applied to the task of semantic video retrieval (word-to-video) task. This task requires user to input a word description and then retrieve videos that have the closet semantic meanings to the input words base on the video contents. The semantic video retrieval task is rarely challenging, as it requires the model to recognize the meaning of unseen words and link it to the most suitable video content. Since lacking of public dataset to evaluate the

produced results, I handmade a word queue pool includes 40 word queries and evaluate their top-10 accuracies based on the representation trained on the UCF101 dataset [25].

The rest of the document is organized as follows.

Chapter II introduces the background theories of the fundamental building blocks I use in this research project. Chapter III describes the proposed semantic spatio-temporal embedding representation approaches in detail. Chapter IV explains the experimental setup, results comparison and analyses. I finally conclude and provide a future work direction in Chapter IV.

CHAPTER 2

THEORY

In this chapter, I will introduce four important algorithms that serve as fundamental building blocks in this work in detail. They are the VGG 16-layer network [12], recurrent neural networks (RNN) and two of its gated alternatives [13,14], the Google *word2vec* word embedding method and the Adadelta gradient descent optimization method [33]; starting from the VGG 16-layer network.

2.1 VGG-16 LAYER NETWORK:

The core idea of VGG-type network is to go deep by stacking small kernels convolution operations (3 × 3) [12] other than going wide by stacking relatively large kernel convolution operation (as the 11 × 11 convolution in the AlexNet [4]). The VGG-16 layer network achieved competitive result (9.5% top-5 error rate) in 2012 ILSVRC ImageNet Classification challenge [10] while maintain small complexity, small memory usage and small weight size (thanks to the thoroughly usage of 3 × 3 kernel convolution layers). Moreover, the network structure is relatively elegant and easy to fine-tune to meet different needs. Hence, the VGG-16 layer network is very popular in serving as a building block (often in the terms of feature extraction) on conducting visual researches. The implementation details and VGG-16 layer structure are described as below.

During training, the input to the VGG-16 layer network is a fixed-size 224 × 224 mean-extracted RGB image. The image is then fed to five steps of layers where each step of layers contains multiple convolution layers. Each convolution layer has receptive field:

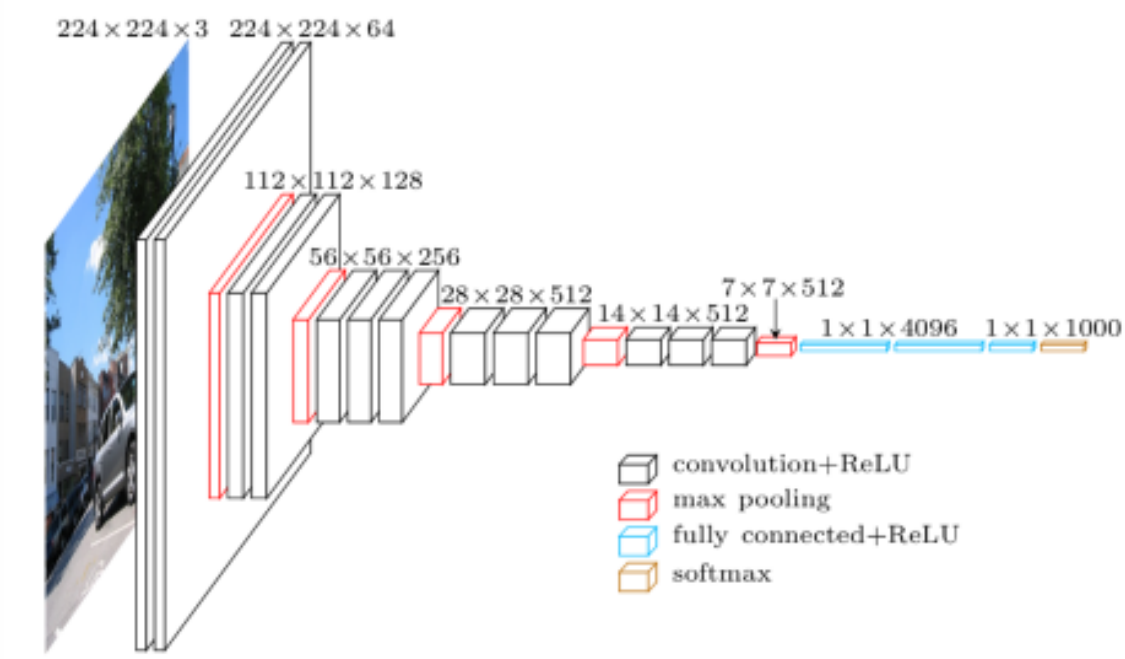

Figure 1. Architecture of the VGG-16 layer network: the network contains 5 steps of layers. The first and second steps contain 2 convolution layers while the rest three steps contain 3 convolution layers. Each convolution layer is equipped with a rectified linear unit (ReLU) to serve as an activation function. Five max-poolings are carried out at the end of every step of layers. Three Fully Connected layers and one softmax layer are implemented at the end to produce output for the ILSVRC classification challenge.

3 × 3 (the smallest size to capture the notion of left/right, up/down and center) with stride 1 pixel and spatial padding 1 pixel to preserve spatial resolution after convolution. Each convolution layer is equipped with a rectified linear unit (ReLU) activation layer. Spatial pooling is carried out by five max-pooling layers, which follow at the end of each step of layers. Max-pooling is performed over a 2 × 2 pixel window, with stride 2 pixels. Three Fully-Connected (FC) layers are implemented at the end. The first two have 4096

channels each; the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The last layer is a softmax layer. The standard VGG-16 layer network structure is shown in figure 1.

The reason why going deep method beat going wide method in the complexity-performance tradeoff is actually simple. For example, to achieve the same receptive field as an 11 × 11 convolution layer (as used in the AlexNet) by stacking 3 × 3 convolution kernels, one needs to stack five 3 × 3 convolution layers. This replacement is actually benefit both complexity-wisely and performance-wisely. An 11 × 11 convolution layer requires 11×11×{input_size}×{output_channels} computation flops while five 3 × 3 convolution layers only requires 5×3×3×{input_size}×{output_channels} computation flops. That is, replace one 11 × 11 convolution layer by five 3 × 3 convolution layers actually give you an around 62.8% reduction in complexity. The same math works for the comparison in weights size. Performance-wisely, stacking five 3 × 3 convolution layers give you more explanation ability as the decision function gains higher power which lead to more non-linear (high-level) features.

In this work, I choose the VGG-16 layer network pre-trained on ImageNet [27] and the VGG-16 layer network pre-trained on the optical flows extracted from YouTube-8M (motion network) [28] to serve as spatial embedding for each RGB frame and optical flow frame in the video clips. For both networks, I pooled the vector output from the second Fully-Connected layer (*fc7*) as spatial embedded features. Both pooled vectors has dimension: 4096×1.

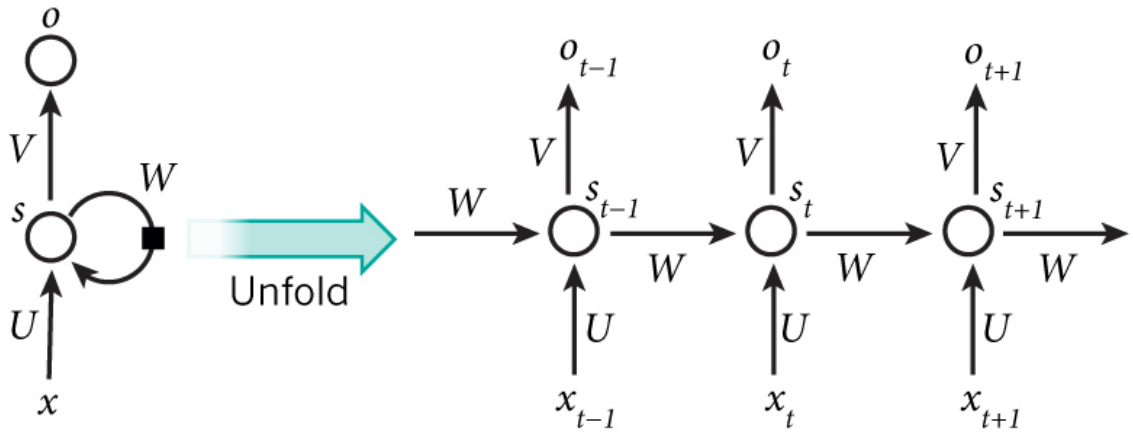

Figure 2. Diagram of the Recurrent Neural Network (RNN) Architecture; the network contains an internal state s and three different types of node: input node U, output node V and hidden node W. For each frame x_t in the video clip x, the network changes his internal state s_{t-1} to s_t according to the input frame x_t. Then, the network produces the output by Hadamard product of the change internal state s_t and the output node.

2.2 RECURRENT NEURAL NETWORK:

The core idea of Recurrent Neural Network (RNN) is to form a directed cycle between its hidden units to create an internal (hidden) state to process arbitrary sequence of inputs. Fig.2 shows a RNN architecture and forwarding computation, where the video clip $x=\{x_1,\cdots,x_t,\}$ is serve as inputs to the RNN one frame at a time through the input node U. Then, the hidden node W changes the internal state s_{t-1} to s_t according to the Hadamard product (element-wise product) of the encoded input and the previous state s_t as:

$$s_t = f(Ux_t + Ws_{t-1}). \tag{3}$$

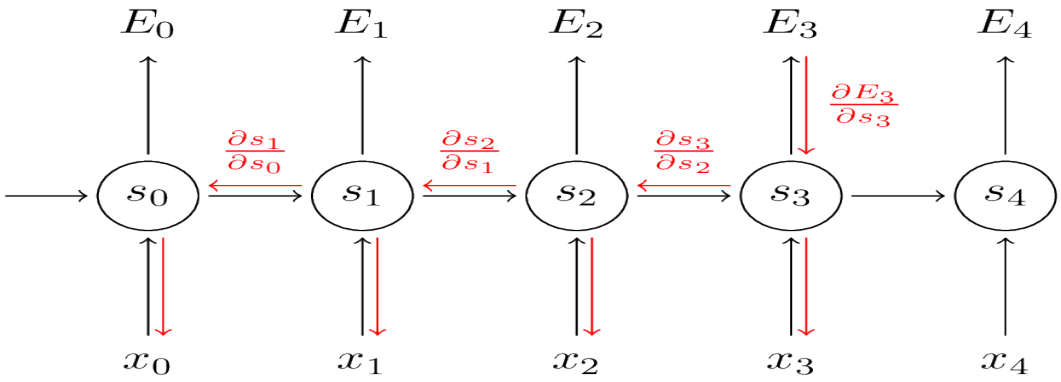

Figure 3. Diagram of the Backpropagation Through Time (BPTT) algorithm: to compute the gradient at $t = 3$, one need to backpropagate the gradient through every previous time steps and sum up all the computed gradients.

The function f is the non-linear activation function (noramlly we use *tanh*). Finally, the output is computed by the Hadamard product of the internal state s and output node V as:

$$o_t = f(Vs_t). \tag{4}$$

Notice that, by repeating the forwarding process for every frames in x, the RNN can unfold the time sequence x and store previous temporal information in his internal state s_t.

Perhaps the most challenge part in RNN is the backward process. If we choose to use the popular Stohastic Gradient Descent (SGD) to learn good weights, since weights in RNN is shared by all time steps in the network, the computed gradient at time t not only depends on the output of time t, but all previous outputs. That is, to compute the gradient at time t, we need to backpropagate every steps previous to t (i.e. t-$1, \cdots 1$) and sum up all their gradients as:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}, \tag{5}$$

where E is the cost function we designed. This is called the Backpropagation Through Time (BPTT) algorithm.

Because of the necessity to go through all previous steps when backwarding, RNN has difficulties in learning good weights for long input sequences due to the chain rule in the BPTT will reduce the gradient whenever it backpropage through a prvious time step. This makes intutitive sence since the non-linear activation function f (tanh or sigmoid) maps all value into range -1 to 1; thus the derivative in BPTT is bounded by 1 as well. This is called the Gradient Vanishing problem as the gradient will vanish progressively in the backpropagation process when the time sequence gets longer. This is very problematic as the network will be biased by recent inputs and choose to forget prevous input which is undesirable for videos where target events can happen at random time. Therefore, in this work, I choose to use the gated-alternatives of RNN to combat the gradient vanishing problem.

2.2.1 Long Short-Term Memory (LSTM):

Long Short-Term Memory (LSTM) is proposed in 1997 [13]. LSTM avoids the gradient vanishing problem by implementing four pairs of gating node $W=\{W_x,W_h,W_c,b\}$ and two internal states – hidden state h and cell state C. Each pair of gating node W contains four gate weights; W_i for input, W_h for hidden state h, W_c for cell state C and bias b. They are applied to serve as the forget-layer f, the input-layer i and the output-layer o respectively to optionally let information to go through and thus empower LSTM to remove or add information to its internal state. This unique feature decreases the gradient

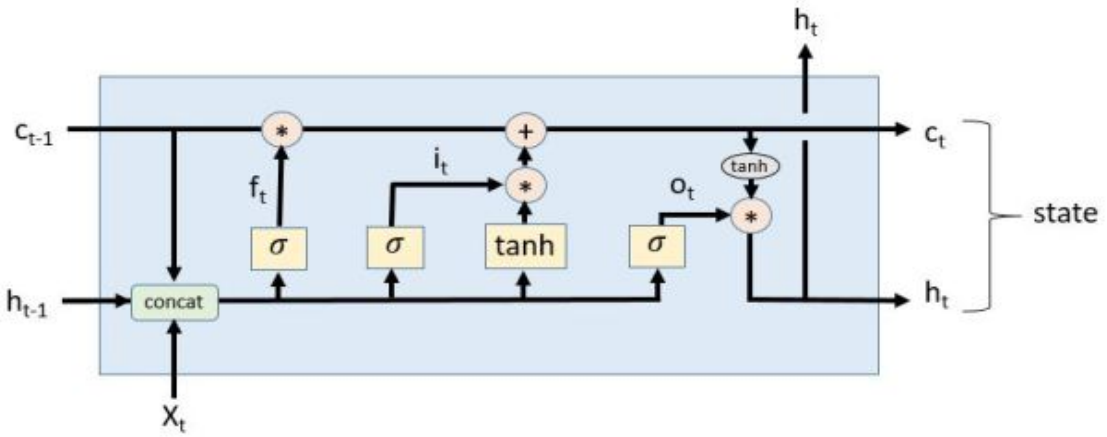

Figure 4. Diagram of the Long Short-Term Memory (LSTM) architecture: LSTM updates the hidden states h_t and cell state C_t by feeding forward previous states h_{t-1} and C_{t-1} along with the current input frame x_t through forget-layer f_t, input-layer i_t the output-layer o_t. These three layers are designed to serve as gating node for the network to preserve global information and drop noise.

vanishing impact by preventing some irrelevant inputs contribute to the current output (the ratio of contribution will determine by the gate values).

Fig.4 shows the architecture of the LSTM. Assuming the same video frame sequence $x=\{x_1,\cdots,x_t,\cdots\}$ as input, at time t, frame x_t is loaded into the network and concated with previous hidden state h_{t-1} and cell state C_{t-1}. Then, the concatenation is used to compute the forget-layer value f_t by:

$$f_t = \sigma(W_{xf} \bullet x_t + W_{hf} \bullet h_{t-1} + W_{cf} \bullet C_{t-1} + b_f). \quad (6)$$

Where $\bullet$ denotes the Hadamard product. Noticed that in LSTM, we normally use logistic sigmoid as activation function σ in order to bound the layer value in the range of 0 to 1 to serve as gate. Meanwhile, the input-layer value i_t is computed by:

$$i_t = \sigma(W_{xi} \bullet x_t + W_{hi} \bullet h_{t-1} + W_{ci} \bullet C_{t-1} + b_i)\,. \tag{7}$$

With both the forget-gate and input-gate computed, we are now ready to compute the new cell state C_t. To update C_{t-1} to C_t, the forget-layer is first applied to the previous cell state C_{t-1} to decide which elements in C_{t-1} should be forgotten/preserved; then the input-layer is applied to the input concatenation as:

$$C_t = f_t \bullet C_{t-1} + i_t \bullet \tanh(W_{xi} \bullet x_t + W_{hc} \bullet h_{t-1} + b_c) \tag{8}$$

Finally, we compute the output-layer value and the hidden state h_t with the help of the freshly obtained cell state C_t as:

$$o_t = \sigma(W_{xo} \bullet x_t + W_{ho} \bullet h_{t-1} + W_{co} \bullet C_t + b_o) \tag{9}$$

$$h_t = o_t \bullet \tanh(C_t) \tag{10}$$

The obtained output h_t (also serves as the hidden state in LSTM) has the same dimension as the input x_t and is embedded to explore global temporal information in the input video clip, which is very desired in video temporal representation.

2.2.2 Gated Recurrent Unit (GRU):

One challenge of the LSTM is the network complexity. A convention Fully-Connected LSTM (FC-LSTM) requires four pairs of weights $W=\{W_x, W_h, W_c, b\}$ where each element in W has the same dimension as the input frame. For example, assume the input sequence is a collection of temporally-related vectors: $v = \{v_1,...,v_n\}$ and each vector has 4096 dimensions (the exact feature size of the *fc7* layer in the VGG-16 layer network). One convention LSTM will have 65536 ($4 \times 4 \times 4096$) weights that needs to be updated. Moreover, for the embedding process (forward pass) you need to consider every time step previous to your current output and each time step has 13 matrix multiplications.

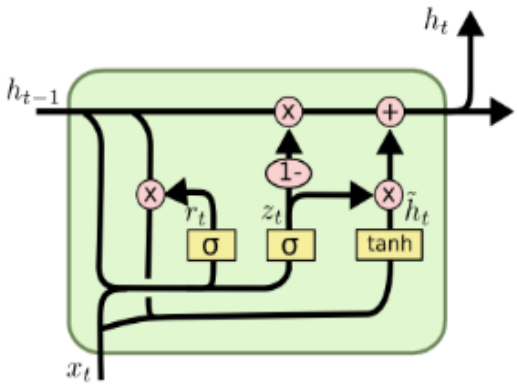

Figure 5. Diagram of the Gated Recurrent Unit (GRU) architecture: GRU simplified the LSTM structure by only maintain one hidden state h_t and three pairs of gating nodes; the reset-layer r_t, the update layer z_t and hidden state proposal h_t.

That is, at time t=n, the network requires n × 12 × 4096 × minibatch computation flops to encode the sequence.

A simpler variation on the LSTM is the Gated Recurrent Unit (GRU) purposed in 2014 [14]. Fig.5 demonstrates a standard Fully-Connected GRU (FC-GRU) structure. When feeding forward, the reset layer r_t is firstly computed by:

$$r_t = \sigma(W_{xr} \bullet x_t + W_{hr} \bullet h_{t-1} + b_r). \tag{11}$$

Where σ, again, is the logistic sigmoid activation function that bound gate values into the range 0 to 1 and • denotes the Hadamard product. The update layer z_t is computed similarly by:

$$z_t = \sigma(W_{xz} \bullet x_t + W_{hz} \bullet h_{t-1} + b_z). \tag{12}$$

Finally we update the hidden state h_t by:

$$h_t = (1 - z_t) \bullet h_{t-1} + z_t \bullet \hat{h}_t, \tag{13}$$

where

$$\hat{h}_t = \tanh(W_{xh} \bullet x_t + W_{hh} \bullet (r_t \bullet h_{t-1}) + b_h) . \quad (14)$$

Noticed that GRU simplifes LSTM by merging the cell state C and hidden state h to one hidden state h. When the reset layer vlaue is close to 0, the hidden state is forced to ignore the previous hidden state and reset with the current input only. Thus allows the hidden state in GRU to drop temporal irrelevants and encode global temporal information. Compared with FC-LSTM, assuming the same input vectors: $v = \{v_1,...,v_n\}$, Complexity-wise speaking, FC-GRU only requires 9 matrix multiplications (FC-LSTM needs 13 matrix multiplications) per time step; therefore enjoys a ~31% reduction in computation flops compared with FC-LSTM. When considering the number of trainable weights, FC-GRU has only 36864 (3 × 3 × 4096) weights, a 43.75% reduction compared with FC-LSTM. This simplication is rather important in high computation demand input such as videos which is why, in this work, I choose FC-GRU to encode frame-level VGG-16 layer network features.

2.3 GOOGLE *WORD2VEC* NETWORK:

The Google *word2vec* newtork [7] aims at mapping each linguistic word to a real-value *k*-dimension vectors in the semantic embedded space. To achive this, the network utililizes the relationship of each target word and its context words by learning from billions of *Wikipedia* documents.

The Google *word2vec* newtork is actually not one particular learning model but a combination of two distinct shallow learning networks, each with two different training methods; Continuous Bag-of-Words (CBOW) with or without negative sampling and skip-gram with or without negative sampling. All the four combinations are capable to

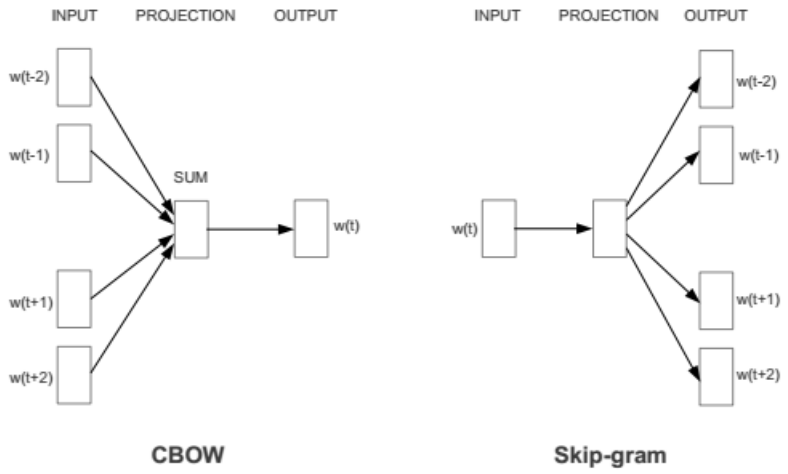

Figure 6: Structure of Continuous Bag-of-Words (left) and skip-gram (right). When training, CBOW predicts the target word *w* given its context *c* while skip-gram predicts the context given the target word.

map each word to a specific vector. In the following paragraph, I will discuss all four combinations starting with CBOW.

Fig.6 (left) shows the structure of a CBOW learning network. CBOW is trained to predict the target word *w* from the contextual words that surround it $\boldsymbol{c}$. Given a corpus *Text*, the goal of CBOW is to learn the weight (parameter) θ so as to maximize the word probability $p(w \mid c;\theta)$:

$$arg\max_{\theta} \prod_{c \in C(w)} [\prod_{w \in W} p(w|c;\theta)] \tag{15}$$

where *C(w)* is the "Bag-of-words" (context of word *w* in *text*) and *W* is the all available words in the dictionary. If using softmax as the classification layer in CBOW, we can rewrite the condition probability to:

$$p(w|c;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{w^i \in W} e^{v_c \cdot v_{w^i}}} \tag{16}$$

where v_w and v_c are the vector representations of the context c and target word w respectively and $\cdot$ refers to the inner-product operation. The learnable weights are the v_w^j and v_c^j for $j=1,\cdots,k$ (k-dimension real value vectors). Notice that by maximize the above equations, we maxmize the inner-product between the v_w and v_c. That is, we transform target word w to a vector v_w in the semantic space by putting it close to the vectors formed by its context v_c.

The idea of skip-gram is actually similar to CBOW but in a reverse way. Fig.6 (right) shows the structure of a skip-gram learning network. While CBOW tries to maximize the word probability $p(w \mid c;\theta)$, skip-gram tries to maximize the context probability $p(c \mid w;\theta)$. That is, given a word w, skip-gram predicts the contextual target words $\boldsymbol{c_t}$ that surround it by learning the weights θ to maximize $p(c \mid w;\theta)$:

$$arg\max_{\theta} \prod_{w \in text} \left[\prod_{c \in C(w)} p(c|w;\theta) \right] \tag{17}$$

This time, *C(w)* is the set of all avialable contexts. Again, by applying softmax as the classification layer in skip-gram, we can rewrite the context conditional probability to:

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c^i \in C} e^{v_w \cdot v_{c^i}}} \tag{18}$$

where v_w and v_c are the vector representations of the context c and target word w respectively and $\cdot$ refers to the inner-product operation. Notice that the training model is not limited to predict only the immediate context, training instances can be created by "skipping a number of gram" and hence the name skip-gram.

Using either CBOW or skip-gram learning network will result in good word embedding v_w in the sense that similar semantic meaning words will be map to similar vectors in the semantic space. However, it is remarkably computation expensive to

maximize either the word probability $p(w \mid c;\theta)$ or the context probability $p(c \mid w;\theta)$ due to the summation $\sum_{w^i \in W} e^{v_c \cdot v_{w^i}}$ over all words in the dictionary (or $\sum_{c^i \in C} e^{v_w \cdot v_{c^i}}$ over all available contexts). Therefore, a more efficient way to deriving word embedding is necessary.

The Google *word2vec* newtork tackles this problem by introduing negative sampling approach. In the following paragraph, I will discuss the negative sampling approach based on skip-gram but notice that the same idea can be easily applied to CBOW (by exchanging the word *w* and context *c* in the conditional probability model).

First, let's introduce an indicator *D* that *D*=1 when the word-context pair (*w*,*c*) comes from the training data *d* and *D*=0 if otherwise. Our goal now is to learn the weights θ to maximize the probabilities that every observation pairs indeed came from *d*:

$$arg\max_{\theta} \prod_{(w,c)\in d} p(D = 1 \mid w, c; \theta) \tag{19}$$

Again, we can replace the probability $P(D = 1|w, c; \theta)$ with softmax

$$P(D = 1|w, c; \theta) = \frac{1}{1+e^{-v_c \cdot v_w}} \tag{20}$$

and rewrite the equation as:

$$arg\max_{\theta} \prod_{(w,c)\in d} \log \left(\frac{1}{1+e^{-v_c \cdot v_w}}\right). \tag{21}$$

By doing this, we have already hugely reduce the computation from summing every possible contexts per each target word to only the word-context pair (*w*,*c*) in the dataset. However, Equation 21 has one trivial solution if we set θ such that $P(D = 1|w, c; \theta) = 1$ for every word-context pair (*w*,*c*). This can be easy complished by letting every vectors representation in the training dataset *d* have the same value ($v_w = v_c$) and have the inner-product of v_w and v_c greater than a positive integer *K*.

To avoid this, we present the model with some word-context pairs (w,c) that are not in the training dataset d ("the negative samples"). Finally, we can add the negative sample term in Equation 21 as:

$$arg\max_{\theta} \sum_{(w,c)\in d} \log\left(\frac{1}{1+e^{-v_c \cdot v_w}}\right) + \sum_{(w,c)\in d'} \log\left(\frac{1}{1+e^{v_c \cdot v_w}}\right), \qquad (22)$$

here d' is the negative samples dataset generating by randomly pairing the word-context pairs in d (assume there are all incorrect pairs). In this work, I choose to use the skip-gram with negative sampling model to semantically embedded the linguistic knowledges to the spatio-temporal video representation because experiements in [7] shows that the skip-gram with negative sampling model represent well in both rare and common words and phrases.

2.4 ADADELTA GRADIENT DESCENT OPTIMIZATION:

Depending on the amount of data, multiple optimization algorithms have been proposed. While the stohastic gradient descent (SGD) optimization method, which update the parameter θ for every training pairs (example $x^{(i)}$ and label $y^{(i)}$), has achieved great success in shallow neural network with relatively small dataset, it may consume a huge amount of time tp stablize when the dataset has high variance. On the other hand, the vanilla gradient descent (or batch gradient descent) optimization method, which update the parameter once for every entire dataset (epoch), is extremely slow against large dataset as it recomputes gradient for similar examples before each parameter update. Vaniila mini-batch gradient descent optimization method finally takes the best of both SGD and vanilla gradient descent by prefroming parameter update every a constant (normally 50~256) amount of training pairs as:

$$\theta = \theta - \eta \bullet \nabla_{\theta} J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) . \tag{23}$$

Where θ is the target parameter, η is the pre-designed learning rate, $x^{(i:i+n)}$ and $y^{(i:i+n)}$ are n training pairs (mini-batch). Vanilla mini-batch gradient descent is popular in nowaday deep learning field as it reduces the varaince of the parameter update (lead to more stable convergence) and can easily applied to parallel computing.

Perhaps the most challenging part in applying vaniila mini-batch gradient descent method is to set a proper learning rate η. A learning rate that is too small can lead to slow convergence or trap in suboptimal local minima, while a large learning rate may hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge. Another major drawback for the pre-designed learning rate in vaniila mini-batch gradient descent is that it cannot adjust itself during training. If the training dataset is sparse and features have large variance in frequencies, the vanilla mini-batch gradient descent cannot weight rarely occuring features so as to perform a larger update to the parameter.

Adadelta [33] addresses the learning rate challenge by storing the gradient information for the past few gradients so that it can dynamically adjust the learning rate based on the gradient difference. Thus able to perform larger updates for infrequent and smaller updates for frequent training samples. Assume $g_{t,i}$ to be the gradient of some objective function J with regard to the parameter θ_i at time t. In the Adadelta update rule, it rewrites Equation 23 to:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[g^2]_{t,i} + \varepsilon}} g_{t,i} . \tag{24}$$

Where $E[g^2]_{t,i}$ is the gradient running average at time t computed by the exponentially decaying average of the previous gradient running average $E[g^2]_{t-1,i}$ and the current gradient $g_{t,i}$ as:

$$E[g^2]_{t,i} = \gamma E[g^2]_{t-1,i} + (1-\gamma) g_{t,i}^2 . \tag{25}$$

Normally, γ is set to around 0.9 (similar to the momentum term) and ε is a small constant to insure the denominator is non-zero. Notcie that in Equation 24, the learning rate η is multiply by the ratio of current gradient and previous gradient running average. If the training sample is rarely seen (infrequent), the ratio will become big which will eventually lead to a larger update for the target parameter θ_i. However, in Equation 24, a default learning rate still need to defined. To overcome this, Adadelta utilize another exponentially decaying average function. This time not of square gradient ratio but of square parameter update as:

$$E[\Delta\theta^2]_{t,i} = \gamma E[\Delta\theta^2]_{t-1,i} + (1-\gamma)\Delta\theta_{t,i}^2 . \tag{26}$$

At this time we can rewrite Eqaution 24 by replacing η with $E[\Delta\theta^2]_{i,j}$ as:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{E[\Delta\theta^2]_{t,i}}{\sqrt{E[g^2]_{t,i} + \varepsilon}} g_{t,j} \tag{27}$$

However, notice that the $(1-\gamma)\Delta\theta_{t,i}^2$ term in Equation 26 is unknown to the update function (actually it is the target to the update function). Adadelta approximates the parameter running average $E[\Delta\theta^2]_{t,i}$ with the previous parameter running average $E[\Delta\theta^2]_{t-1,i}$ and finally yields the Adadelta update rule:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{E[\Delta\theta^2]_{t-1,i}}{\sqrt{E[g^2]_{t,i} + \varepsilon}} g_{t,j} \tag{28}$$

or equivalently,

$$\theta_{t+1} = \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \tag{29}$$

Where RMS stands for the root-mean square function. Notice that using the Adadelta update rule (Equation 28), one do not even need to have a default learning rate setup [33].

In this work, I choose to apply the Adadelta gradient descent optimization to train our end-to-end architecture because of its robustness in finding good local minima and fast convergence.

CHAPTER 3

METHOD

In this chapter, I will introduce my proposed semantic spatio-temporal embedding so as to support a set of popular semantic analysis tasks. The proposed approach encodes the underlying video clips sequentially in the order of spatial, temporal and semantic embedding, respectively.

3.1 SEMANTIC SPATIO-TEMPROAL EMBEDDING:

To embed spatial information of the given video, I employ VGG-16 layer network [12] as the basic building blocks to extract frame-level spatial features for its demonstrated performance in exploring spatial correlations. To capture the temporal structure of the given video, one may utilize Recurrent Neural Network (RNN) on top of the frame-level spatial features. However, basic FC-RNN is known to suffer from the vanishing gradient problem especially when the input time sequence goes longer. Two well-known alternatives are the Long-Short Term Memory (LSTM) [13] and Gated Recurrent Unit [14] (GRU), which avoid gradient vanishing by implementing "gating units" that decide what to "forget" and what to "update". In our approach, I choose FC-GRU to further encode frame-level spatial features, since they have comparative performance to FC-LSTM while requiring less computation cost, which is an important consideration for video analysis.

Furthermore, recognizing that the learned vector representations, although rich in spatio-temporal information, still lack a semantic organization or clustering that would

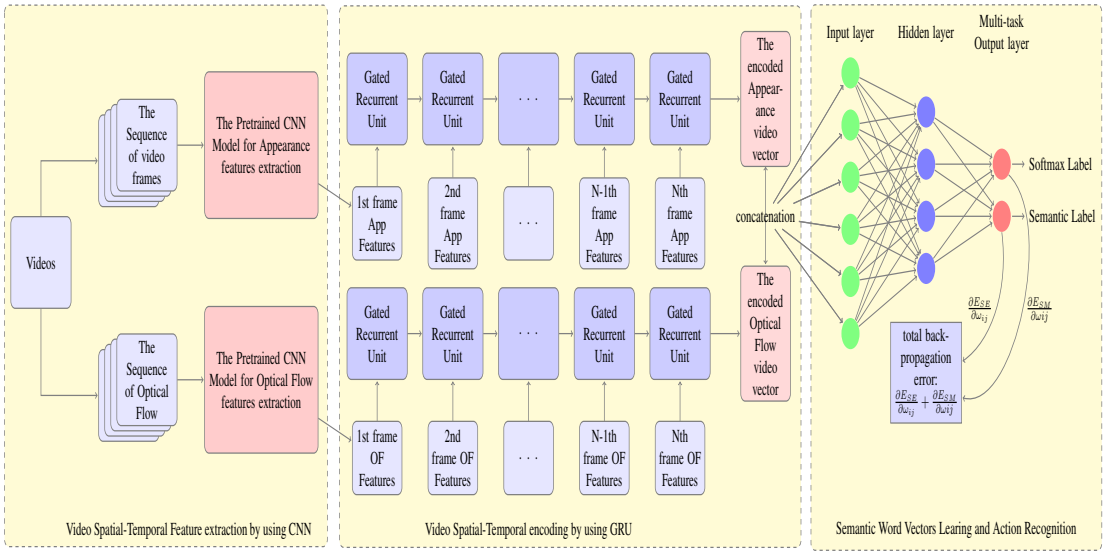

Figure 7. Architecture for the proposed semantic spatio-temporal embedding: RGB and optical flow features are forwarded into two different GRUs; outputs are simply concatenated; then fed into a two-output fully-connected multilayer perceptron (MLP); One output is binary to suit action recognition purpose while another produce the desired embedding for high-level analysis task.

directly facilitate a higher-level analysis task like action labeling, I propose to learn an additional mapping from the learned vector representations of the videos to the *word2vec* (skip-gram with negative sampling) Semantic Embedding Space (SES) [7]. This learning task relies on labels of the videos in a training set (data-driven) and the SES learned from *Wikipedia* documents with more than 1 billion words (prior knowledge on semantic meanings of the labels). Hence the final mapping effectively leads to an embedding for the videos into the SES, enabling the utilization of the word (label) semantics for any higher-level analysis task. The overall framework is illustrated in figure 7.

3.1.1 Spatial Features and Motion Features:

With a collection of video clips V where each clip $v \subset V$ contains a sequence of frames with a specific temporal order $\{f_1,..,f_n\}$ and label l_v. Inspired by the two-stream convolution network [5], I pull out both RGB frames to represent spatial information and compute the dense optical flows from the frame sequence to represent (local) motion information. Both the dense optical flows and the RGB frames are processed at 10fps, and the optic flows are computed by using the implementation described in [29].

Two distinct pre-trained VGG-16 layer network models are then used to extract appearance f_{v_app} and optical flow features f_{v_of}. The VGG-16 layer network [12] pre-trained model on the ImageNet ILSVRC-2012 dataset [10] is responsible for extracting appearance features while the pre-trained networks implemented by [28] is used to extract optical flow features. I collect f_{v_app} and f_{v_of} from the last Fully-Connected layer (*fc7*) from each pre-trained model as the spatial embedding video vector sequences. Both have dimension: 4096×1.

3.1.2 Fully-Connected GRU-based Temporal Embedding:

Given the spatial embedding video vector sequences f_{v_app} and f_{v_of}, I encode each of them independently with two variable-length FC-GRUs. The choice of encoding separately is based on the hypothesis that f_{v_app} and f_{v_of} contain different and/or complementary types of information of the video at different space-time scale and thus they should not be pooled together at the frame level.

The Fully-Connected Gated Recurrent Unit (FC-GRU) uses the reset gate r_t and the update gate z_t (both gates take values between 0 and 1 due to the logistic sigmoid

activation function) to fuse input with previous memory and define how much of the previous memory to keep, thus avoiding the gradient vanishing problem while maintaining the power to discover temporal correlation. Given an input sequence $x = \{x_1,..,x_t,...,x_N\}$, the FC-GRU encoder forwards the input by iterating the following equation from $n = 1$ to N :

$$\begin{aligned} r_t &= \sigma(W_{xr} \bullet x_t + W_{hr} \bullet h_{t-1} + b_r) \\ z_t &= \sigma(W_{xz} \bullet x_t + W_{hz} \bullet h_{t-1} + b_z) \\ \hat{h}_t &= \tanh(W_{xh} \bullet x_t + W_{hh} \bullet (r_t \bullet h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \bullet h_{t-1} + z_t \bullet \hat{h}_t \end{aligned} \tag{30}$$

where h_t is the model hidden state at step t, $\hat{h}_t$ is the proposed hidden state update at step t, $\sigma(\bullet)$ denotes the logistic sigmoid function, $\bullet$ denotes Hadamard product (element-wise) and W_{xr}, W_{xz}, W_{xh}, W_{hr}, W_{hz}, W_{hh}, b_r, b_z, b_h are the FC-GRU hidden weights.

The FC-GRU architecture I use in this project contains 1024 hidden units for each FC-GRU hidden weight (the dimension of h_t is 1024). I experimented with various numbers of units and chose 1024 to best trade off computation expense and performance.

I then pooled the last hidden state h_N as outputs for both RGB frames and optical flows sequence. The outputs of our FC-GRU encoder are the appearance video representation E_{app_v} and the optical flow video representation E_{of_v}, both having 1024 dimensions. Then, I perform a normalization that set the energy for both E_{app_v} and E_{of_v} to 1. Finally, a simple concatenation is performed to combine the appearance and optical flow representations at the video level: $Ev = [E_{app\_v}; E_{of\_v}]$. This results in a temporally encoded representation Ev of 2048 dimensions.

3.1.3 Semantic Embedding Using Fully Connected MLP:

In order to embed semantic knowledge to the encoded video representation *Ev*, a Fully Connected MLP (FC-MLP) is trained to serve as the mapping function $g(\bullet)$ that projects *Ev* to its corresponding coordinate *g(Ev)* in the *word2vec*-transformed semantic embedding space. Here the *word2vec*-transformed 500-dimensional semantic embedding space is trained by skip-gram with negative sampling [7] on billions of *Wikipedia* documents. Dimension of word vector representation is set to 500 to tradeoff training complexity and maintaining semantic relationship [7].

Semantic label vectors $m = \{m_1,..,m_N\}$ are obtained by feeding forward the given video labels $l = \{l_1,..,l_N\}$ through the pre-trained skip-gram with negative sampling network as: $m_n = M(l_n)$. For labels that contain multiple words, such as "riding horse" or "rock climbing indoor", I generate a single vector z_n by averaging the word vectors for all unique words in the label l_n:

$$\mathrm{l_n} : m_n = \frac{1}{w} \sum_{w \subset \mathrm{l_n}} M(\mathrm{l_n}) . \tag{31}$$

Two distinct loss functions are implemented and errors from both loss functions are backpropagated to train the end-to-end model. To embed semantic label vectors m_n, I use the hinge rank loss function with margin *mar* as:

$$loss_1 = -\sum_{j \neq n} \max[0, mar - m_n \bullet g(E_v) + m_j \bullet g(E_v)] . \tag{32}$$

Notice that, a normalization process is applied to the mapped video vectors $g(E_v)$ that set the energy of $g(E_v)$ to 1 before loss is computed. The hinge rank loss function requires $g(E_v)$ to be more similar (in the sense of cosine similarity since $g(E_v)$ is normalized) to their corresponding label vector m_n than other counterparts m_j by a pre-set constant margin *mar*. The second loss function serves as a fine-tuning part and is effective when

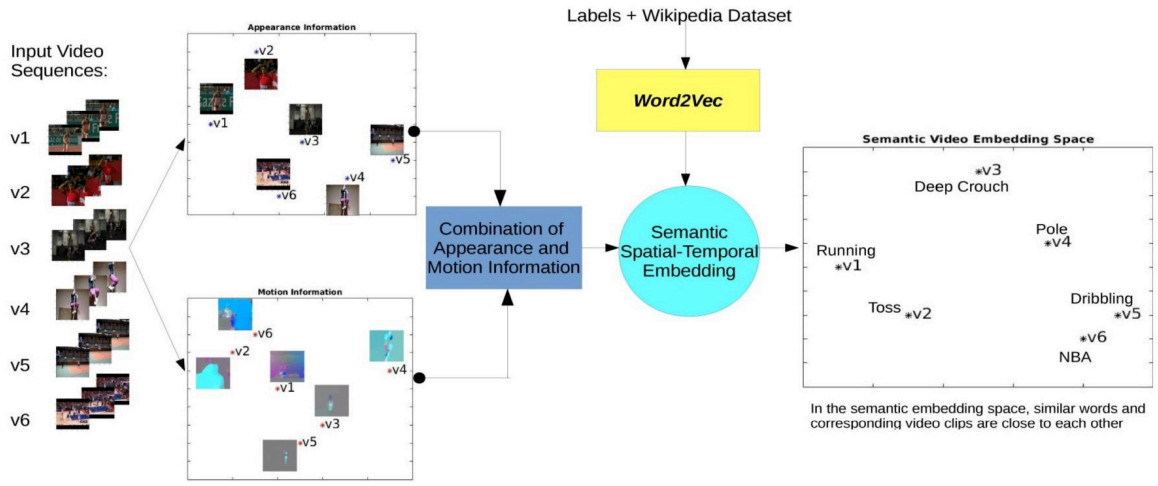

Figure 8. Illustrating the working of the proposed model. Videos carry similar semantic meanings can vary greatly in terms of spatio-temporal features (e.g., videos v6 "NBA" and v5 "Dribbling" are far from each other when only temporal encoding is performed). The global temporal structure of these features is encoded by GRUs. A learned mapping further embeds the GRU-encoded spatio-temporal feature into a semantic embedding space, where diverse videos sharing the similar semantics cluster together (e.g., "NBA" and "Dribbling" videos are projected to similar coordinate after embedding).

performing video recognition task. It measures the probability of $g(E_v)$ belonging to its corresponding label l_v through a softmax layer. The second loss function is defined as:

$$loss_2 = -\sum_{n=1}^{N} \{l_v = n\} \log(P(l_v = n \mid g(E_v)))$$
$$P(l_v = n \mid g(E_v)) = \frac{e^{g(E_v) \bullet w_v}}{\sum_{n=1}^{N} e^{g(E_v) \bullet w_n}} \quad , \tag{33}$$

where the indicator function $\{l_v = n\}$ equals to 1 when the video clip label l_v belongs to the n category and 0 otherwise and w is the softmax layer weights. Notice that, both

softmax and semantic output leverage visual and semantic similarity to train the Fully-Connected MLP since two outputs share information in the FC-MLP hidden layers. Moreover, the summation of softmax and semantic loss is backpropagated to learn weights in each layer, so that each loss can be served as a compensation and fine-tuning part for the other.

Fig. 8 illustrates how the proposed model works for semantically encoding a video. Video clips with similar semantic information may still be dramatically differently in the spatio-temporal domain, which is supposed to be captured by the VGG-16 layer network and GRU networks. Upon the coding of the GRU network, the spatio-temporal representation of the video is embedded into the *word2vec* semantic space, where video clips sharing similar semantic meanings would be close by, and the learned mapping via FC-MLP will enable processing/analyzing new videos without labels (or whose labels were never used in training the mapping).

3.2 *HUBNESS* PROBLEM IN ZERO-SHOT CLASSIFICATION:

Although the learned Fully-Connected MLP mapping function can introduce the human linguistic knowledge to the video representations, it, meanwhile, introduces another challenge – the "*hubness*" problem [30].

The *hubness* problem happens when the video dataset used on training is so much smaller (not general enough) compared with the *word2vec* semantic embedding space (which contains billions of distinct word vectors). In this case, the Fully-Connected MLP mapping function learned from the limited amount of video-word vector pairs will be biased as it only capable of mapping video vectors to a small area in the semantic space

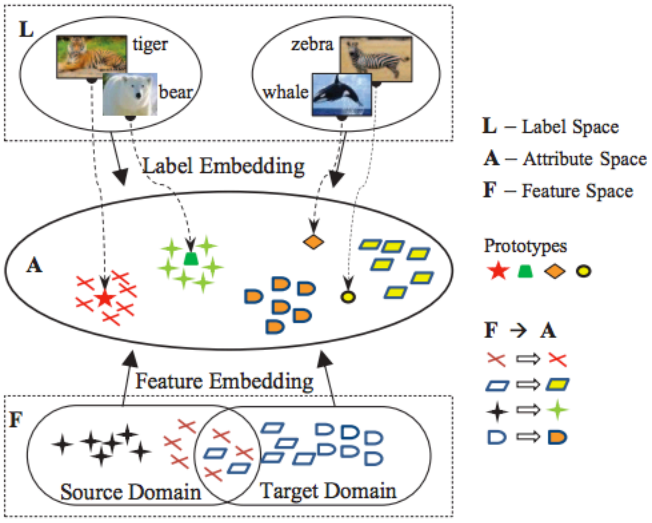

Figure 9. The *hubness* problem. When the training video dataset is insufficient (or not general) enough compared with the *word2vec* label embedding semantic space. The overlap for the training source domain and testing target domain may be very small or even none, which lead to dramatic accuracy drop when performing zero-shot classification. Figure reference from [31].

(source domain), which lead to small or none overlap between the training source domain and the testing target domain when testing. Fig. 9 demonstrates the small or none overlap between source domain and target domain when the source domain is trained by a relatively small amount of data. This problem will hugely affect the zero-shot classification accuracies, as there may have no trained reference vectors exist around the mapped test video vectors in the semantic space at all. In this work, I implemented a new strategy called "Convex Combination of Similar Semantic Embedding Vectors" (ConSSEV) to tackle this problem.

3.2.1 Convex Combination of Similar Semantic Embedding Vectors:

The Convex Combination of Similar Semantic Embedding Vectors (or ConSSEV) utilizes the benefits of both semantic and softmax outputs in our model to adjust the mapped test video vectors $g(E_v^t)$ to a new prototype $g^*(E_v^t)$. For semantic output, I first find the top K training label vectors m_k where $k \in$ training dataset and $k = 1, \dots, K$ that have the highest cosine similarity scores with the mapped test video vector $g(E_v^t)$. Then I form a new prototype $g^*(E_v^t)$ by weighted sum of all K vectors with their corresponding softmax probabilities $p(l_v = k|g(E_v^t)$ where $k \in$ training dataset as:

$$g^*(E_v^t) = \frac{1}{K}\sum_{k=1}^{K}\left[\left(l_v = k\middle|g(E_v^t)\right) \cdot m_k\right]. \quad (34)$$

ConSSEV avoids the *hubness* problem by replacing the test video vectors $g(E_v^t)$ with a new prototype $g^*(E_v^t)$ formed by weighted sum K trained label vectors. Thus "shift" the test video vectors into the small hyper-plane i.e. the train domain.

CHAPTER 4

EXPERIMENTS AND RESULTS

In this chapter, I will discuss the experiments designed to evaluate our semantic spatio-temporal embedding. I will start from introducing the dataset then explain some parameters I pre-set to the model training and finally analyze results for three distinct experiments (video action recognition, video zero-shot classification, and semantic video retrieval) and make discussions for each of them.

4.1 UCF101 ACTION RECOGNITION DATASET:

I train and test our model on the well-known UCF101 Action Recognition dataset [25], for its diversity of contents as well as the ready availability of results from many state-of-the-art baselines. UCF101 contains 13320 real-life videos from 101 caption categories collected from YouTube and has the largest action diversity in terms of viewpoint, scale, environment condition⋯etc. Thus make it one of the most challenging dataset in video analysis. Videos in each action category are grouped into 25 groups where each group shares some common features, such as background, viewpoints, objects...etc. Fig. 10 shows the action categories for the entire dataset. For training and testing purposes, I split the UCF101 dataset in three different ways so as to meet the need of three distinct experiments.

For measuring the video recognition results, I use the standard splits (“Three Train/Test Splits”) for UCF101 dataset provided by the official website [25]. For each split, the model takes 66.6% of the dataset as training and test on the rest 33.3%. I then

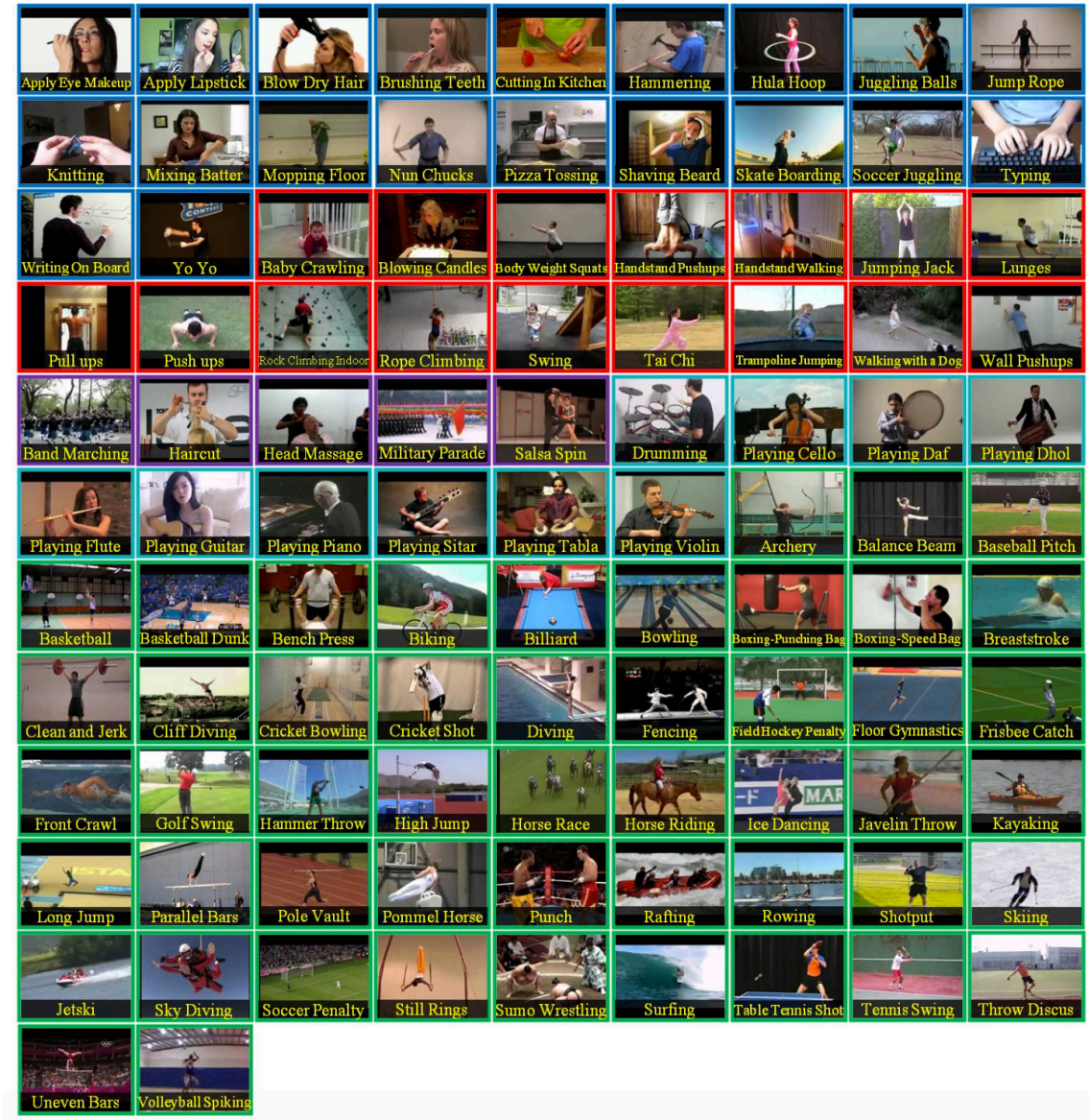

Figure 10. The UCF101 action recognition dataset [25]: 13320 real-life video clips from 101 categories; each category contains 25 group, where each group can consist 4~7 videos of a specific action. There are five major types of actions in the dataset: 1. Human-Object Interaction, 2. Body-Motion, 3. Human-Human Interaction, 4. Playing Musical Instrument and 5. Sports. The large diversities in the dataset make UCF101 one of the most challenging dataset in video analysis.

average the accuracy of each split to compute the mean accuracy.

For measuring the zero-shot classification result, I use the same evaluation protocol as in [32] – 30 independent splits for UCF101 dataset with each split contains a completely disjoint 51 categories for training purpose and 50 for testing purpose. I then average the accuracy of each split to compute the mean accuracy.

For measuring the semantic video results, I follow the first train-test split rule in [25]. For the part of queue word pool, since there are no existing benchmark dataset, I manually crafted 40 queue words to serve as the input queue. Words in the queue word pool are not limited to action names (such as *Hiking* and *Hunting*) but also include some adjectives (such as *Extreme* and *Classical*), organization/country names (such as *NBA*, *FIFA* and *India*) and celebrity names (such as *Beethoven* and *Yankees*). A complete table of the handmade query pool can be found in table 3.

4.2 TRAINING MODEL SETUP:

All three tasks are solved by the proposed semantic spatio-temporal embedding architecture. Dense optical flow and RGB frames are extracted at 10fps. Spatial features for both RGB and optical flow frames are extracted at the last Fully-Connected layer (*fc7*) in VGG-16 layer network with 4096 dimension. Each FC-GRU unit contains 1024 hidden units. The FC-MLP has 2048, 1024 and 500 nodes in its input, hidden and output layers respectively. The margin for hinge rank loss computation, however, varies between tasks: 0.4 for zero-shot classification and 0.55 for video recognition and semantic video retrieval. In this work, I choose the Adadelta [33] as the gradient descent optimization algorithms for its demonstrated performance in finding good convergence in gradient

Algorithm	**Accuracy (%)**
Dense Trajectory (Wang *et al*) [3]	75.1
LRCN (Donahue *et al*) [7]	82.9
Composite LSTM model (Srivastava *et al*) [16]	84.3
Two-stream Convolution Net (Simonyan *et al*) [5]	88.0
Deep LSTM with 30 Frame (Ng *et al*) [18]	88.6
TDDs (Wang *et al*) [15]	**91.5**
Our result (softmax)	83.9

Table 1: 3-fold video recognition accuracy on the UCF101 dataset

descending. The γ value is set to 0.9 and learning rate for the whole end-to-end structure is initialized as 0.0001 for the first 15 epochs and then reduced by half for every 15 epochs. The mini-batch size is set to 30 videos per mini-batch.

4.3 THREE EXPERIMENTS:

I evaluate the proposed video representation architecture on UCF101 dataset [25] by conducting three visual tasks: video action recognition, video zero-shot classification, and semantic video retrieval. All three tasks are solved by using the proposed semantic spatio-temporal embedding architecture.

4.3.1 Video Recognition:

Video recognition task is performed as a proof point for the proposed representation successfulness in encoding the temporal and spatial information in videos. Video recognition task takes raw videos as input and classify the given video to one of the categories (one of the 101 categories in UCF101). In this wok, when training, I follow

the previous mentioned three train-test split rule. When testing, I categorize test videos to the trained label that has maximum probability based on the softmax layer output.

I compare the result performance of the proposed semantic spatio-temporal embedding with [3] [5] [7] [15] [16] and [18] as shown in Table 1. The proposed semantic spatio-temporal embedding demonstrates competitive results against other methods. Compared with [5], [15], our video representation does not require late classifier-based fusion or pooling method on multiple sources of features (including both handcrafted features and learning model based features). Compared with [18], our video representation does not train on multi-layer FC-LSTM and thus is more time efficient. I believe that applying late fusion on multiple sources of features and multilayer FC-GRU can potentially lead to further improvements. However, in this work, I focus on demonstrating that our video representation can handle much higher-level task.

4.3.2 Zero-shot Classification:

Zero-shot classification has always been treated as one of the most challenging task in video analysis since the lacking of available references (labeled examples) for test videos. Zero-shot classification takes raw videos as input and uses the trained model to classify test videos whose categories have never been presented to the learning model (hence unseen). In this work, I performed the 30 independent splits protocol mentioned in [32]. For each split, 51 out of the 101 categories are served as training data and the rest 50 categories are used to evaluate the trained model. When testing, I categorize test videos to the ”unseen” test label that has maximum cosine similarity (nearest neighbor) to our temporal and semantic embedded representation.

Algorithm	Accuracy (%)
ConSE (Norouzi *et al*) [34]	10.5
SES (Xu *et al*) [32]	10.9
IAP (Lampert *et al*) [26]	12.8
DAP (Lampert *et al*) [26]	14.3
SES with Self-training (Xu *et al*) [32]	15.8
Our result	**14.7**
Our result with ConSSEV	**28.8**

Table 2: 30-fold zero-shot classification accuracy on the UCF101 dataset

I compare our results with [26], [32] and [34] as shown in Table II. Without any domain-shift strategies, the proposed semantic spatio-temporal representation outperforms the state-of-the art attribute-based IAP and DAP methods by 2% and 0.5% respectively which indicate the effectiveness of the proposed semantic embedding technique that encodes semantics as well as spatio-temporal information. Moreover, notice that the proposed semantic spatio-temporal representation enjoys a huge boost in accuracy by applying the ConSSEV domain-shift strategy to tackle the *hubness* problem. It significantly outperforms [32] which also tackle the *hubness* problem by finding new test prototypes in the training video vectors but without any softmax prior knowledge about the model uncertainty.

4.3.3 Semantic Video Retrieval:

To further demonstrate that the proposed representation can perform "semantic association" of videos, I challenge it with the semantic video retrieval task. For this task, I follow the first train-test split rule in [25] to separate the UCF101 dataset. A word

Query Labels	Top10 Retrieve Results	Query Labels	Top10 Retrieve Results
NBA	Basketball Dunk (10)	Extreme	Rock Climbing Indoor (5), Uneven Bars (2), Soccer Juggling (2), Pole Vault (1)
Orchestra	Playing Cello (9), Playing Piano (1)	Tide	Cliff Diving (4), Surfing (2), Throw Discus (2), Sky Diving (1), Rafting (1)
Army	Military Parade (10)	India	Paying Tabla (4), Playing Sitar (2), Head Massage (1), Cricket Shot (1), Mixing (1)
Music	Playing Sitar (9), Playing Piano (1)	Celebrate	Military Parade (6), Long Jump (1), Band Marching (1), Ice Dancing (1), Blowing Candles (1)
Computer	Typing (10)	Home-run	Baseball Pitch (5), Basketball Dunk (3), Field Hockey Penalty (1), Frisbee Catch (1)
Park	Biking (9), Golf Swing (1)	Boat	Kayaking (4), Rafting (2), Rowing (2), Cliff Diving (1), Push Ups (1)
Summit	Cliff Diving (7), Skiing (2), Rope Climbing (1)	Toy	Yo-yo (4), Nun chucks (4), Pull Ups (1), Juggling Ball (1)
School	Skate Boarding (10)	Snow	Skiing (2), Ice Dancing (2), Cricket Bowling (1), Pole Vault (1), Blowing Candles (1), Blow Dry Hair (1), Rafting (1), Sky Diving (1)
Park	Biking (9), Golf Swing (1)	Acrobatics	Juggling Balls (5), Soccer Juggling (5)
Water	kayaking (10)	Ocean	Cliff Diving (4), Sky Diving (3), Kayaking (2), Rafting (1)
FIFA	Soccer Penalty (8), Soccer Juggling (2)	Hurl	Throw Discus (2), Mopping Floor (2), Baby Crawling (1), Javelin Throw (1), Cricket Shot (1), Blowing Candles (1), Pull Ups (1)
Club	Golf Swing (8), Soccer Juggling (2)	Hiking	Biking (5), Kayaking (4), Rafting (1)
Nature	Tai Chi (7), Hammering (2), Walking with Dog (1)	Swim	Diving (5), kayaking (3), Cricket Bowling (1), Sky Diving (1)
Beethoven	Playing Cello (8), Playing Voilin (2)	Jogging	Biking (5), Skate Boarding (2), Soccer Juggling (1), Skiing (1), Ice Dancing (1)
Classical	Playing Cello (7), Playing Voilin (3)	Foam	Blowing Candles (7), Pull Ups (1), Rope Climbing (1), Juggling Balls (1)
Yankees	Baseball Pitch (10)	Hip-hop	Trampoline Jumping (6), Swing (4)
Duel	Boxing Punching Bag (8), Punch(2)	Scramble	Pull Ups (6), Trampoline Jumping (2), Rope Climbing (1), Cricket Shot (1)
Lifting	Body Weight Squats (4), Rope Climbing (4), Pull Ups (2)	Mat	Rope Climbing (4), Pommel Horse (3), Trampoline Jumping (2), Javelin Throw (1)
Martial	Fencing (3), Archery (3), Boxing Punching Bag (3), Balance Beam (1)	Parachuting	Diving (6), Cricket Bowling (2), Hand Stand Walking (1), Sky Diving (1)
Tumbling	Trampoline Jumping (8), Throw Discus (1), Frisbee Catch (1)	Hunting	Horse Riding (3), Kayaking (3), Nun chucks (3), Frisbee Catch (1)

Table 3: Query word pool and their retrieval results: Query words are never seen in training dataset. Retrieve results are the retrieved video clips denoted by their categories in UCF101 dataset. Number in the brackets denoted how many video clips belong to this category are retrieved in the top10 hit list. Notice that, the input query word is not limited to action category (such as *Hunting* and *Jogging*) but can be any type of word (such as *Extreme* and *Yankees*).

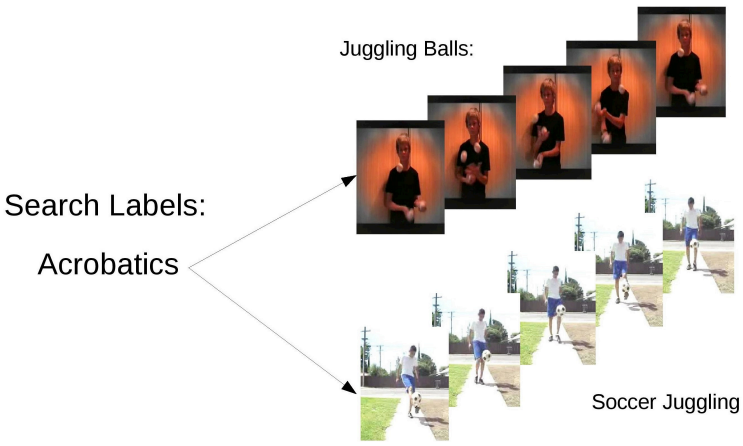

Figure 11. Examples of the proposed architecture performing semantic video retrieval task; By inputting the query word "Acrobatics", the proposed model retrieves "Juggling Balls" and "Soccer Juggling" video clips.

pool, which contains 40 words, was created manually to serve as query words. When testing, I first feed forward all test videos through the trained architecture to obtain their semantic embedding representation. Then, for each query word, the model retrieves top 10 test video clips that have maximum cosine similarity (nearest neighbor) to the *word2vec* transformed query word as demonstrated in figure 11. Note that, for a query, which contains multiple words, I perform the same averaging method as described in Equation 31.

The query word pool and their retrieval results are shown in Table 3. It shows that the proposed representation is capable of capturing both visual contents and their semantics. One specific example is the retrieved results using the query word "NBA". All retrieved videos belong to the "Basketball Dunk" category in the UCF101 dataset. Another more complex example is "Acrobatics", which leads us to "Juggling Balls" and

"Soccer Juggling" videos. However, since the correctness of this task is very subjective and rarely been performed, I cannot find a way or standard benchmark to objectively quantize the performance of the proposed model when performing the semantic video retrieval task.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this chapter, I will conclude this work and discuss an extension alternative for the proposed video representation method. The extension alternative aims to simultaneously encode the spatial and temporal information by implementing spatial local convolution kernels for the GRU architecture (called Conv-GRU) to replace the traditional Fully-Connected GRU so as to increase the spatio-temporal representative in video vectors.

5.1 CONCLUSION:

Methods that provide meaningful and efficient video representations that support high-level video vision tasks are in high demand. While most of the previous works in this area take handcrafted features or pool/fuse frame-level spatial encoded features to represent videos, this research project introduced an elegant end-to-end video representation architecture that learn semantic spatio-temporal embedding for videos sequentially.

To fully utilize the semantic and spatio-temporal nature of videos, the proposed architecture sequentially embeds the spatial, temporal and semantic knowledge to form the video representation. Two VGG-16 layer pre-trained networks are applied to extract high-level spatial features for RGB and optical flow frames. Then two Fully-Connected Gated Recurrent Units are utilized to encode the extracted features temporally and

separately. Finally a Fully-Connected Multilayer Perceptron is trained to embed human linguistic knowledge to the spatio-temporal video vector form the video label vector.

The proposed video representation is evaluated on the UCF101 dataset for action recognition, zero-shot classification and semantic video retrieval. The action recognition result demonstrates that the proposed approach is able to provide useful and efficient video representations for global information analysis tasks. In zero-shot classification task, the proposed embedding outperforms the state-of-the-art attribute-based method and set a new benchmark in the UCF101 dataset. Moreover, a novel domain-shift method called ConSSEV is proposed to tackle the *hubness* problem in zero-shot classification and further boost the representation performance. Finally, in the semantic video retrieval task, the proposed approach demonstrates its ability in retrieving semantically meaningful videos simply based on a simple word query.

Overall, this work is an attempt at trying to learn better features so as to support high-level video analysis tasks. Results show that the proposed architecture not only produces quality global features but also equipped with semantic word knowledge. As part of future work, I try to deal with the major drawback of this approach: the Fully-Connected GRU fail to encode local spatial information.

5.2 FUTURE WORK:

Although the proposed representation is embedded with spatial, temporal and semantic information, I argue that the representation did not fully explore the spatial and temporal knowledge in the given videos. Thus cannot beat the state-of-the-art method (multiple features late-fusion) [15] in action recognition task. This lack of spatio-

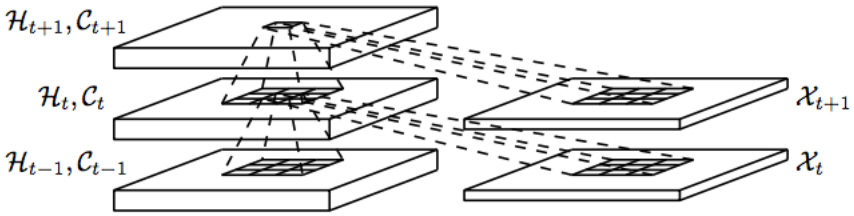

Figure 12. Inner structure of ConvLSTM. The ConvLSTM determines the future state of both cell layer C_{t+1} and hidden layer h_{t+1} in the grid by the current inputs, previous states and its local neighbors with the help of convolution kernels [35].

temporal representative comes from the usage of full connections in input-to-state and state-to-state transactions in the Fully-Connected GRU (thus fail to encode any spatial information) and the usage of *fc7* features in the VGG-16 layer network (which only contains the global spatial information). Therefore, to deal with the spatial and local nature of video frames, I proposed an alternative method to encode the spatial and temporal information simultaneously by replacing the Fully-Connected GRU by multiple stacks of spatial convolution GRU (ConvGRU) as the future work of this research project.

Fig.12 demonstrate the inner structure of a similar technique – the ConvLSTM that was recently proposed in Shi et al for precipitation nowcasting and next frame prediction [35]. In their work, they replaced the fully connected kernels in the FC-LSTM by either 3 × 3 or 5 × 5 convolution kernels as:

$$
\begin{aligned}
f_t &= \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \bullet C_{t-1} + b_f) \\
i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \bullet C_{t-1} + b_i) \\
C_t &= f_t \bullet C_{t-1} + i_t \bullet \tanh(W_{xi} * x_t + W_{hc} * h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \bullet C_t + b_o) \\
h_t &= o_t \bullet \tanh(C_t)
\end{aligned}
\tag{35}
$$

Model	Number of parameters	Cross entropy
FC-LSTM-2048-2048	142,667,776	4832.49
ConvLSTM(5x5)-5x5-256	13,524,496	3887.94
ConvLSTM(5x5)-5x5-128-5x5-128	10,042,896	3733.56
ConvLSTM(5x5)-5x5-128-5x5-64-5x5-64	**7,585,296**	**3670.85**
ConvLSTM(9x9)-1x1-128-1x1-128	11,550,224	4782.84
ConvLSTM(9x9)-1x1-128-1x1-64-1x1-64	8,830,480	4231.50

Table 4: Accuracy and Complexity comparison between Fully-Connected LSTM and ConvLSTM. Number in ( ) denotes the convolution kernel size while number after – refers to the number of filters in the convolution filter banks.

Where • refers to the Hadamard product (element-wise product), * denotes the convolution operations and, again, σ is the logistic sigmoid function. They demonstrated in the experiments [35] that by applying this replacement, the model captures better spatio-temporal correlation and achieves lower cross entropy loss in the generated Moving-MNIST dataset [16]. Meanwhile, this replacement can help save computation expense because the elimination of the usage of expensive Fully-Connected computations. Table 4 shows the cross-entropy loss and complexity improvement by this replacement in predicting the next frame in the Moving-MNIST dataset [35].

The core idea of my proposed spatio-local convolution GRU (or ConvGRU) is similar to the ConvLSTM. However, I decide not to completely eliminate the usage of pre-trained VGG network as [35] so as to better trade-off complexity and performance. Instead, I plan to leave the first convolution step (first two 3×3 convolution layers, their ReLU activation layers and the 2×2 max-pooling layer) in the VGG network as these two layers are known to be low-level edge detection filters and Gabor filters which are general enough for most video frames. The output from the VGG network has dimension:

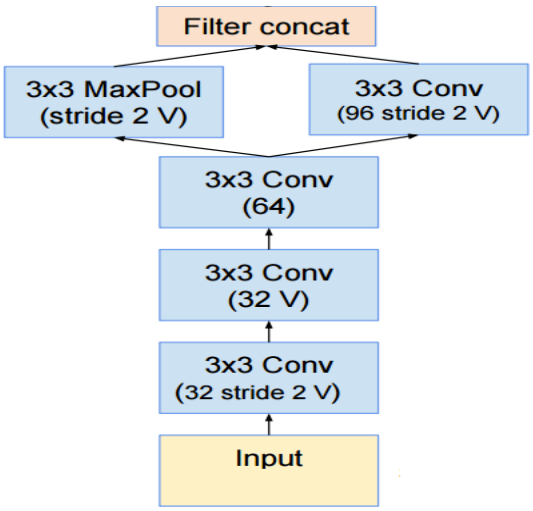

Figure 13. Part of the stem schema in Google Inception-v4. The input has dimension: 16×112×112 and the output has dimension: 192×27×27. "V" denotes no spatial padding and max-pooling has kernel size 3×3.

128 × 112 × 112 (assume the given video frame has dimension: 3 × 224 × 224) which is too expensive if directly feed into the ConvGRU. Hence, I decide to build a 1 × 1 convolution layer without activation layer to serve as a channel activation layer and reduce the channel from 128 to 16 (16 × 112 × 112) before forwarding to the proposed ConvGRU architecture. The proposed ConvGRU with channel activation encoded the local spatial and temporal information as:

$$
\begin{aligned}
\overline{x_t} &= f_{1x1}(x_t) \\
r_t &= \sigma(W_{xr}(\overline{x_t}) + W_{hr} \bullet h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}(\overline{x_t}) + W_{hz} \bullet h_{t-1} + b_z) \\
\hat{h}_t &= \tanh(W_{xh}(\overline{x_t}) + W_{hh} \bullet (r_t \bullet h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \bullet h_{t-1} + z_t \bullet \hat{h}_t
\end{aligned}
\quad . \qquad (36)
$$

Where $f_{1x1}(\cdot)$ is the 1 × 1 convolution layer with no activation which serve as channel activation. Notice that, $W_{xr}(\ )$, $W_{xz}(\ )$ and $W_{xh}(\ )$ are not single convolution kernel but stacks of multiple convolution layers with x_t as input. Such stacks can be implemented as some part of any well-known structures. I decide to use the part of the stem architecture described in the Google-inceptionv4 [36] (shown in figure 13) as it performs well empirically.

REFERENCES

[1] Laptev, Ivan. "On space-time interest points." International Journal of Computer Vision 64, no. 2-3 (2005): 107-123.

[2] Klaser, Alexander, Marcin Marszałek, and Cordelia Schmid. "A spatio-temporal descriptor based on 3d-gradients." In BMVC 2008-19th British Machine Vision Conference, pp. 275-1. British Machine Vision Association, 2008.

[3] Wang, Heng, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. "Action recognition by dense trajectories." In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 3169-3176. IEEE, 2011.

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[5] Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." In Advances in Neural Information Processing Systems, pp. 568-576. 2014.

[6] Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A neural probabilistic language model." journal of machine learning research 3, no. Feb (2003): 1137-1155.

[7] Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[8] Lowe, David G. "Object recognition from local scale-invariant features." In Computer vision, 1999. The proceedings of the seventh IEEE international conference on, vol. 2, pp. 1150-1157. Ieee, 1999.

[9] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." In European conference on computer vision, pp. 404-417. Springer Berlin Heidelberg, 2006.

[10] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. "Imagenet large scale visual recognition challenge." International Journal of Computer Vision 115, no. 3 (2015): 211-252.

[11] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9. 2015.

[12] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014).

[13] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

[14] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[15] Wang, Limin, Yu Qiao, and Xiaoou Tang. "Action recognition with trajectory-pooled deep-convolutional descriptors." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4305-4314. 2015.

[16] Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhutdinov. "Unsupervised learning of video representations using lstms." CoRR, abs/1502.04681 2 (2015).

[17] Donahue, Jeffrey, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. "Long-term recurrent convolutional networks for visual recognition and description." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2625-2634. 2015.

[18] Yue-Hei Ng, Joe, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. "Beyond short snippets: Deep networks for video classification." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4694-4702. 2015.

[19] Li, Haojie, Lijuan Liu, Fuming Sun, Yu Bao, and Chenxin Liu. "Multi-level feature representations for video semantic concept detection." Neurocomputing 172 (2016): 64-70.

[20] Gitte, Madhav, Harshal Bawaskar, Sourabh Sethi, and Ajinkya Shinde. "Content based video retrieval system." International Journal of Research in Engineering and Technology 3, no. 06 (2014).

[21] Veltkamp, Remco, Hans Burkhardt, and Hans-Peter Kriegel, eds. State-of-the-art in content-based image and video retrieval. Vol. 22. Springer Science & Business Media, 2013.

[22]Snoek, Cees GM, and Marcel Worring. "Concept-based video retrieval." Foundations and Trends in Information Retrieval 2, no. 4 (2008): 215-322.

[23] Venugopalan, Subhashini, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. "Sequence to sequence-video to text." In Proceedings of the IEEE International Conference on Computer Vision, pp. 4534-4542. 2015.

[24] Venugopalan, Subhashini, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. "Translating videos to natural language using deep recurrent neural networks." arXiv preprint arXiv:1412.4729(2014).

[25] Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." arXiv preprint arXiv:1212.0402 (2012).

[26] Lampert, Christoph H., Hannes Nickisch, and Stefan Harmeling. "Attribute-based classification for zero-shot visual object categorization." IEEE Transactions on Pattern Analysis and Machine Intelligence 36, no. 3 (2014): 453-465.

[27 Chatfield, Ken, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Return of the devil in the details: Delving deep into convolutional nets." arXiv preprint arXiv:1405.3531 (2014).

[28] Gkioxari, Georgia, and Jitendra Malik. "Finding action tubes." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 759-768. 2015.

[29] Brox, Thomas, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. "High accuracy optical flow estimation based on a theory for warping." In European conference on computer vision, pp. 25-36. Springer Berlin Heidelberg, 2004.

[30] Dinu, Georgiana, Angeliki Lazaridou, and Marco Baroni. "Improving zero-shot learning by mitigating the hubness problem." arXiv preprint arXiv:1412.6568(2014).

[31] Kodirov, Elyor, Tao Xiang, Zhenyong Fu, and Shaogang Gong. "Unsupervised domain adaptation for zero-shot learning." In Proceedings of the IEEE International Conference on Computer Vision, pp. 2452-2460. 2015.

[32] Xu, Xun, Timothy Hospedales, and Shaogang Gong. "Semantic embedding space for zero-shot action recognition." In Image Processing (ICIP), 2015 IEEE International Conference on, pp. 63-67. IEEE, 2015.

[33] Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

[34] Norouzi, Mohammad, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S. Corrado, and Jeffrey Dean. "Zero-shot learning by convex combination of semantic embeddings." arXiv preprint arXiv:1312.5650 (2013).

[35] Xingjian, S. H. I., Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." In Advances in Neural Information Processing Systems, pp. 802-810. 2015.

[36] Szegedy, Christian, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, inception-resnet and the impact of residual connections on learning." arXiv preprint arXiv:1602.07261 (2016).